1. **WebApi\_Handson**

**Objectives:**

1. Explain the concept of RESTful web service, Web API, and Microservice

RESTful web services follow the architectural principles of REST (Representational State Transfer), which is an approach that uses standard HTTP methods such as GET, POST, PUT, and DELETE for communication between client and server. RESTful services are stateless, meaning each request is independent and does not rely on previous interactions. Messages in REST are typically exchanged in formats like JSON or XML, but REST is not restricted to XML and can support multiple content types. A Web API is a framework in .NET that allows the development of HTTP-based services. It is lightweight, easy to use, and supports content negotiation, allowing responses in multiple formats such as JSON and XML. In contrast to traditional SOAP-based web services, Web APIs are more suitable for mobile and web applications. A microservice is an architectural style where an application is divided into small, independently deployable services that focus on specific business capabilities. Each microservice can be developed, deployed, and scaled independently, making the system more modular and maintainable.

2. Explain what is HttpRequest and HttpResponse

HttpRequest and HttpResponse are key components of communication between the client and the server in a web application. An HttpRequest represents the incoming request from the client and contains essential information such as the HTTP method (GET, POST, etc.), headers, query strings, URL, and body data. It provides the server with all the details needed to process the request. HttpResponse, on the other hand, is the server’s response back to the client. It includes the status code (e.g., 200 OK, 400 Bad Request), response headers, content type, and the actual response data. These objects form the foundation of client-server interaction in Web APIs.

3. List the types of Action Verbs used in Web API

Action verbs in Web API represent the operations that can be performed on a resource. These include HttpGet, HttpPost, HttpPut, and HttpDelete. The HttpGet verb is used to retrieve data from the server, while HttpPost is used to submit new data to the server. HttpPut is used to update existing data, and HttpDelete is used to remove data from the server. In Web API, these verbs are declared as attributes above action methods within a controller. For example, [HttpGet] is placed above a method that fetches data, ensuring the method responds only to GET requests. This clear mapping between verbs and operations enhances readability and maintainability.

4. List the types of HttpStatusCodes used in Web API

Web API uses various HTTP status codes to indicate the outcome of a request. Common status codes include 200 OK, 400 Bad Request, 401 Unauthorized, and 500 Internal Server Error. A 200 OK response means the request was successful and the server is returning the requested data. A 400 Bad Request indicates that the client has sent invalid or malformed data. The 401 Unauthorized status signals that the request requires authentication, while a 500 Internal Server Error means something went wrong on the server side. These status codes are returned using action result types like Ok(), BadRequest(), Unauthorized(), and StatusCode(500, "Error") in the controller methods.

5. Demonstrate the creation of a simple Web API with Read and Write actions

A simple Web API can be created by defining a controller that inherits from ApiController or ControllerBase. Within this controller, methods are defined for each CRUD operation using the corresponding HTTP verb attributes. For example, a [HttpGet] method retrieves all items, [HttpPost] adds a new item, [HttpPut] updates an existing item, and [HttpDelete] removes an item. Each method returns an appropriate HTTP response using action results. The Web API structure typically includes a Controllers folder containing API controllers, a Program.cs or Startup.cs file to configure services and middleware, and model classes representing the data. This structure allows clean separation of concerns and easy scalability.

6. Explain the types of configuration files in Web API

In .NET Core Web API, configuration files include Startup.cs, appsettings.json, and launchSettings.json. The Startup.cs file is used to configure services and middleware in the application using dependency injection. It defines the request pipeline and endpoint mapping. The appsettings.json file stores configuration data such as connection strings and application settings in JSON format. The launchSettings.json file contains debug profile settings such as environment variables and the application’s launch URL. In .NET Framework (4.5), Web API uses files like WebApiConfig.cs and RouteConfig.cs to define routing for APIs and MVC controllers, respectively. The Web.config file is used to manage application settings, database connections, and system configurations. These configuration files are essential for defining the behavior and structure of the Web API.

1. **First Web Api using .Net core**

**ValuesController.cs**

using Microsoft.AspNetCore.Mvc;

namespace FirstWebApiDemo.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ValuesController : ControllerBase

{

// GET: api/<ValuesController>

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

// GET api/<ValuesController>/5

[HttpGet("{id}")]

public string Get(int id)

{

return "value";

}

// POST api/<ValuesController>

[HttpPost]

public void Post([FromBody] string value)

{

}

// PUT api/<ValuesController>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody] string value)

{

}

// DELETE api/<ValuesController>/5

[HttpDelete("{id}")]

public void Delete(int id)

{

}

}

}

**Output:**

